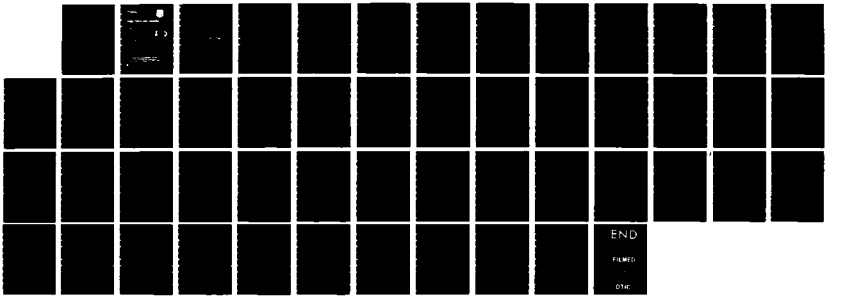
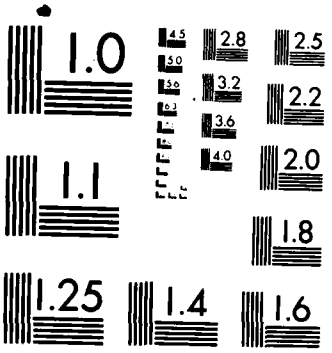
AD-A162 383 OPTICAL SYSTOLIC ARRAY PROCESSORS(U) AERODYNE RESEARCH INC BILLERICA MA H J CAULFIELD ET AL SEP 85 ARI-4657 RADC-TR-85-162 F30602-84-C-0130 1/1

UNCLASSIFIED F/G 9/2 NL





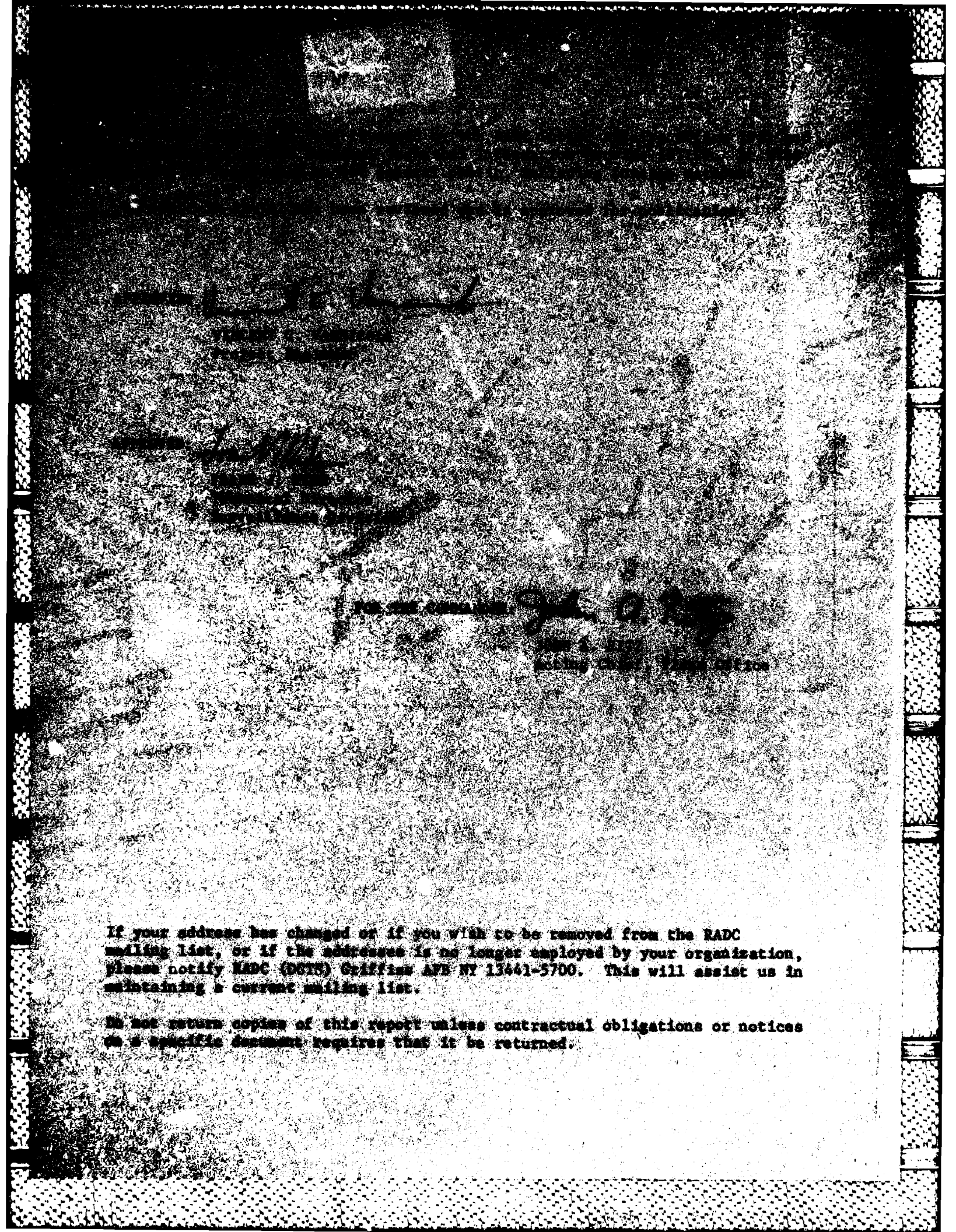
OPTICAL SYSTOLIC ARRAY PROCESSORS

Aerodyne Research, Inc.

H. J. Caulfield, Scott Lyrest and Roger Putnam

This effort was funded totally by the Laboratory Directors' Fund

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700



If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (DGTS) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A162 383

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS N/A
2a SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) ARI-4657	5 MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-85-162
6a NAME OF PERFORMING ORGANIZATION Aerodyne Research, Inc. 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Rome Air Development Center (OCTS)
6c ADDRESS (City, State, and ZIP Code) 45 Manning Road Billerica MA 01821	7b ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a NAME OF FUNDING / SPONSORING ORGANIZATION Rome Air Development Center 8b OFFICE SYMBOL (If applicable) OCTS	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-84-C-0130
8c ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 61101F; PROJECT NO: LDFP; TASK NO: 01; WORK UNIT ACCESSION NO: C4

11 TITLE (Include Security Classification)
OPTICAL SYSTOLIC ARRAY PROCESSORS

12 PERSONAL AUTHOR(S)
H.J. Caulfield, Scott Israel, Roger Putnam

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM Jun84 TO Jun85	September 1985	60

16 SUPPLEMENTARY NOTATION
This effort was funded totally by the Laboratory Directors' Fund (19

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Optical signal processing, and
09	03		Computer architecture OE
17	09		

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Multiplication with 16 bit accuracy can be accomplished by optical means and is competitive and superior in speed to fully electronic devices if a sum-of-products algorithm is being implemented. The advantage of optics is not in the multiply function where electronic multipliers will be generally faster due to the extra analog-to-digital converters needed in the optical device. The optics advantage appears when many of the resulting numerical products need to be added which is accomplished by integrating sequential light pulses onto one detector. The bottleneck for digital addition is the arithmetic carry terms which are evaluated sequentially. We also present and analyze a threshold logic analog-to-digital converter intended for the optical multiplier which demonstrates increased speed over a successive approximation A/D without the complexity of a flash converter. We also present a significant step which is the sideways summer. The sideways summer greatly reduces the time required for the one step of electronic addition which is part of the digital-multiplication by analog-convolution (Cont'd)

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL VINCENT C. VANNICOLA	22b TELEPHONE (Include Area Code) (315) 330-4437
	22c OFFICE SYMBOL RADC (OCTS)

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Block 20 (Continued)

algorithm. This electronic addition is needed to convert the resulting "mixed binary" form of the output into a normal binary number.

TABLE OF CONTENTS

1. WHY USE OPTICS

There have been several reasons why optics would be a suitable choice for an adaptive signal processor. First, and foremost has been the speed and bandwidth of electro-optic devices which can be used. Detectors and sources have bandwidths of greater than 5 GHz, and are commercially available. Second, these electro-optic devices are normally run with milliwatt power levels such that the speed/power factor can provide a significant advantage over electronics. Also we would like to note, that while many people claim optics has an EMI advantage over electronics, it must be realized that the electro-optic devices are driven by the very electronic devices that are prone to EMI effects. Therefore, we will not claim EMI advantages since an optical processor is at present still largely composed of electronics.

2. OPTICAL ARRAY PROCESSORS

For the past two decades, optics has shown great processing power by using its parallel interconnect strength. For instance, a Fourier Transform of an object can be produced as quickly as the rise time of the detector and source. Since each point in the Fourier Plane is a combination of every point in the image plane, a system that has 64K (256 x 256) points in its image has 4×10^9 interconnections performed simultaneously.

There have been two problems associated with optical processors: 1) lack of a fast spatial light modulator with a large number of pixels, and 2) accuracy. Both problems are addressed in this report. However in this work we have assumed that the signal processing would be digital because optics can provide fast calculations, but is limited by the dynamic range of the system, especially the input and output sources and detectors. Sources are usually LEDs or laser diodes and both can be modulated by current. However, LED's obey an exponential law; that is the output power is related exponentially to the input current. This makes the LED extremely difficult to linearize. Most optical systems assume a dynamic range of 30-40 dB. While this is suitable for many problems, the adaptive phased array radar problem requires 2 to 3 times that range. After all, antenna patterns can be created that have sidelobes down 40-50 dB, so an adaptive processor must have at least that range. Hence, optics must do its calculations digitally if it is to be effective. Ten bits of accuracy provides 60 dB of dynamic range, and is well within the current range of A/D converters.

3. DIGITAL OPTICS

Optics can perform digitally by using the Digital Multiplication by Analog Convolution (DMAC) algorithm, which has been in use over the past 6 years. This is based on the fact that if two binary numbers are convolved, and these results are A/D'ed, then shifted to the left and added, the answer is the same as if the two numbers were multiplied digitally, as shown below.

```
  111          111 * 111 = 12321
x 111              1 -  A/D          01
-----              2 -  A/D         10X
  111              3 -  A/D        11XX
 111               2 -  A/D       10XXX
111                1 -  A/D      01XXXX
------                           ------
110001                           110001
```

4. HISTORY

To see how the DMAC algorithm is best implemented, it would be beneficial to briefly look at the history of Optical Signal Processing.

Before the DMAC algorithm, all OSP was done in the analog domain. However, the DMAC algorithms allowed the accuracy to increase to any number of bits as needed with an increased expenditure of time. For example, we previously said that the multiplying (analog) of two numbers optically is done within the risetimes of the source and detector. To digitally multiply two numbers takes 2N-1 clock cycles (to convolve two N bit numbers uses 2N-1 products and sums). Since the clock speed for acousto-optic devices can be from 500 MHz to 1 GHz, this multiplication can be done in about 50-100 ns for a 16 bit multiplication. See Figure 1. However, this result is in "mixed

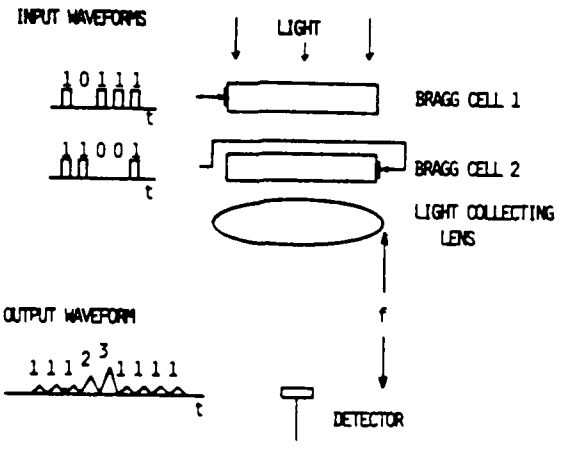


Figure 1.

binary" or analog form; it then must be converted to binary by an A/D converter and a sum shifter which is a special digital adder (this will be addressed in detail later in the report). However, this digital addition part of the process can take an additional 100 ns. Compared to current robust electronic multipliers (TRW data book), which use 50-100 ns to produce the entire digital product, this optical algorithm is feeble. However, five years ago when it was invented, it was clearly as good, if not better than the electronic multipliers.

Fortunately, optics did not stop its development either. The DMAC algorithm was extended to incorporate one of optic's great strengths; parallelism. For example, by placing several acousto-optic cells side by side and imaging the same source across them, it is possible to do many multiplications simultaneously. As multichannel A.O. cells become available, parallel multiplication becomes feasible. For instance, instead of doing one product in 100 ns, it becomes possible, by using a 32 channel A.O. cell, to do 32 multiplications in 100 ns.

In 1981, P. Guilfoyle[1] demonstrated a new architecture by using 2 crossed multichannel A.O. cells. This SAOBic (see Figure 2) architecture has proved to be semi-optimized for calculating matrix vector products. By using crossed multichannel cells, this architecture utilizes the DMAC algorithm, the parallelism of optics, and the common nature of many large processing problems to create a powerful array processor. Since the important adaptive processing problem is essentially a question of matrix inversion, and we have discussed how optics solves this problem, one would assume the adaptive processing problem is solved. On the periphery this may seem true, but when one examines the problem and its solution in detail, several significant weaknesses of Bulk Acousto-Optic processing using the DMAC algorithm come to the surface. These will be evaluated and discussed in the following section.

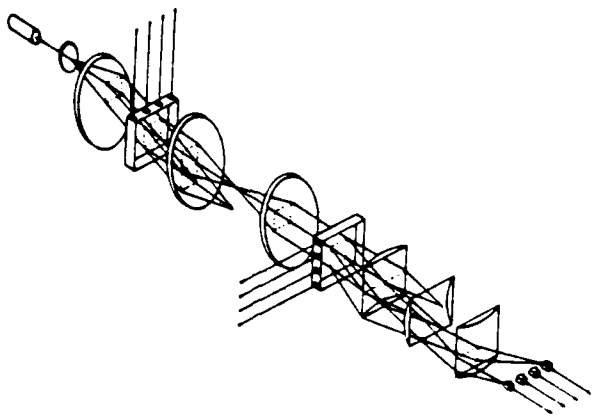


Figure 2. Digital accuracy AO vector-matrix multiplier. Two multitransducer Bragg cells are imaged onto each other.

5. DETAILED LOOK AT DMAC

There are three basic steps involved in the Digital Multiplication by Analog Convolution algorithm.

1. Form a mixed binary product by convolution,
2. Pass through an A/D converter,
3. Shift and add.

All three of these can be improved upon.

1. The convolution of 2 sequences each of N words long uses 2N-1 clock cycles. Therefore, this is one limiting factor in producing a product. As devices get faster, the speed can improve, but considering that A.O. cells operate with GHz bandwidth, this improvement would be marginal. A large improvement would come from decreasing the number of clock cycles needed.
2. A/D converters limit the conversion speed from mixed binary to binary.
3. Improve the Shift and Add (SA) routine as this process could take the longest amount of time.

Finally, although a great number of signal processing problems can be reconfigured into a matrix problem format, there are problems associated with radar signal processing that can be processed more efficiently with a more general modular array. For example FFT's, perfect shuffles, etc. are not optimized for the matrix format.

It was Aerodyne's intention to examine these three fundamental problems of digital optical computing, propose alternatives, and integrate this hardware with algorithms that solve the adaptive radar problem.

5.1 Solutions to the DMAC Algorithms

5.1.1 Asynchronous Integrated Optical Multiply Accumulate Unit

5.1.1.1 Introduction

Current bulk acousto-optic processors calculate products very rapidly yet they suffer the following limitations: 1) the data is clocked into the processor thereby limiting the ultimate speed of the processor to that of the clock, 2) the processor is limited to matrix calculations, 3) an extremely stable, large platform is needed for the apparatus, and 4) a long time is needed to convert the mixed binary output to useable (at least to the rest of the computer world) pure binary.

The first three limitations will be addressed here, and we will propose an alternative design. The fourth problem is currently under extensive investigation by many researchers.

5.1.1.2 Asynchronous Processors

The first limitation (that of having a clocked processor) was recognized by electronic designers more than a decade ago. The earliest designs of digital electronic multipliers used synchronous (clocked) logic. As the designs became more sophisticated, circuitry involving only asynchronous logic evolved such that present designs use a clock only for the input and output of data. These multipliers are built out of half and full adders (AND gates and EXCLUSIVE-OR gates). The time it takes for the multiplication of two n bit words is on the order of n times the propagation delay time of a full adder.

Figures 3a and 3b show our design for an integrated optical asynchronous multiply unit. The two data words to be multiplied are input to the multiply unit. This multiply unit is basically a set of AND gates. However, the output of each multiply is summed such that the partial products (there are 2n-1 partial products for a n x n bit multiply) are represented by a "mixed binary output." Thus a mixed binary result is common to most optical computers. Note that the time for this multiplication is essentially the

$$
\begin{array}{ccccccccc}
 & & & & y_0 & & y_1 & & y_2 \\
 & & & & x_0 & & x_1 & & x_2 \\
\hline
 & & & & x_2 y_0 & & x_2 y_1 & & x_2 y_2 \\
 & & & x_1 y_0 & & x_1 y_1 & & x_1 y_2 & \\
 & & x_0 y_0 & & x_0 y_1 & & x_0 y_2 & & \\
\hline
 & s_0 & & s_1 & & s_2 & & s_3 & s_4
\end{array}
$$

Figure 3a. Flash (Asynchronous) Multiplier Scheme

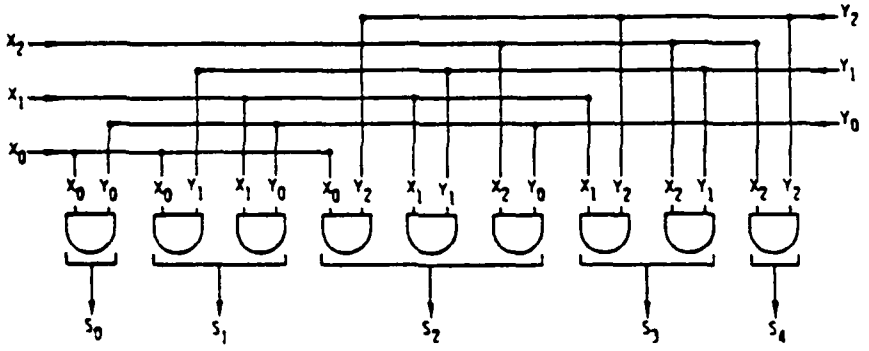


Figure 3b. Flash Multiplier Integrated Optics Implementation

switching time of the electro-optic device (0.1 ns for $LiNbO_3$; 100 ps for GaAs etc.) plus the response time of the detector (less than 1 ns). Therefore, the entire multiplication with a mixed binary result occurs in about 1 ns with current electo-optic technology. In contrast the acousto-optic technology needs 2n-1 clock cycles (which amounts to about 62 ns for a 16 bit multiply with a 500 MHz acousto-optic cell). Furthermore, the proposed multiplier needs only one switching cycle regardless of the size of the data word. This makes the device very attractive for precise (e.g. large data words) calculations.

While this multiplier creates a mixed binary output in a very short time, a comparatively long time is needed to convert the mixed binary output to pure binary. When this conversion time is added to the multiply time, it seems that at its very best, the optical multiplier is not much faster than an off-the-shelf electronic component (why change the technology for a factor of two improvement?).

While multiplication is an important process; it is not enough. Something is usually done with the product; frequently addition. FFT's, convolutions, matrix and vector-vector products all come under the process described as "sum of products" (SOP). (For this reason multiply accumulate units have been developed.)

It so happens that the strength of optics is the weakness of digital electronics. This process is addition; optics does it at the speed of light, electronics does it at the propagation delay of a full adder (3-5 ns for VLSI, 10 ns for LSI). Figure 4 shows the timing needed for an asynchronous integrated-optical multiply and accumulate unit. The workings of this device are extremely simple. Instead of sending the mixed binary result to the A/D converter, the value is held in a sample and hold circuit. The next two data words are sent through the multiplier and their product is then added to the previous one.

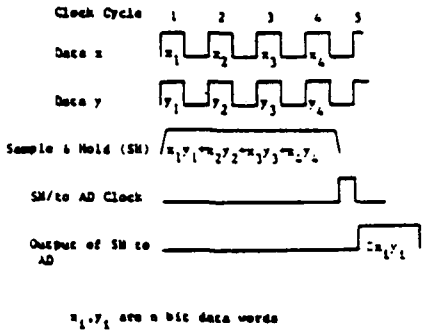


Figure 4. Timing For Multiply Accumulate Unit

For example, we may want to repeat the process so that we sum four products. Each cycle takes approximately 2 ns, (1 ns to load the data, another for the data to pass through the multiply unit). In this manner we can calculate the total sum in mixed binary in 8 ns. Now the data is sent through the associated circuitry to convert the mixed binary to pure binary. Since this output circuitry accounts for 95% of the time used in the calculation (a multiply occurs in 1 ns, it is converted to binary at most in 25 ns), the advantage of the optical accumulator is appearing. Off-the-shelf electronic multiply accumulate units need 160 ns to perform 1 multiply accumulate (640 ns for four 32 bit products) whereas our proposed optical device requires only 60 ns.

5.1.1.3 Matrix Calculations

The second limitation (being restricted to matrix calculations) is not really a limitation since many large computational problems are or can be setup in matrix format. Unfortunately, there are some common computational problems (FFTs) that cannot be setup in matrix format. This inflexibility is solved with the integrated optical unit. It is used at the register level, rather than at the processor level (which limits acousto optic processors to matrix processors). Therefore any calculations involving multiplies, or multiply-accumulates can be solved using the integrated optic device (or several of them) in place of current electronic or optical devices. Also, the integrated optic device keeps some of the parallelism of 3D optics; it sums many partial products upon one detector simultaneously, yet it retains the flexibility (e.g. being able to be placed on a circuit board and being used in a variety of architectures) of electronic chips.

5.1.1.4 Mechanical Stability

The final limitation is solved by realizing that this integrated optical module can be made pin for pin compatible with current electronic chips. The size of a 16 bit multiply unit using current integrated optical technology would be about 3" x 1". This is nearly the same size as a standard all electronic 16 bit multiply chip. Compared to a small optic bench, especially where spot sizes are on the order of 10-50 μm, we feel that our device offers inherent advantages such that engineering time spent on stabilizing the optical platform could be better spent on architectures and algorithms.

5.1.2 Analog To Digital Conversion Using Threshold Logic Elements

This report describes the operation of a threshold A/D converter and compares the accuracy and speed required of its components to those of other A/D converter designs.

5.1.2.1 Operation of the Threshold A/D

The elements of this threshold A/D are analog comparators and Digital to Analog (D/A) converters. The threshold logic element is the analog comparator which sums several inputs, each with different weights and compares the weighted sum to a reference voltage which is the threshold. The inputs to the comparators are the input analog signal and the digital outputs of other comparators. The various weighted inputs to the comparators which are driven by logic signals comprise, in effect, several D/A converters.

Figure 5 shows a 2-bit threshold A/D converter.

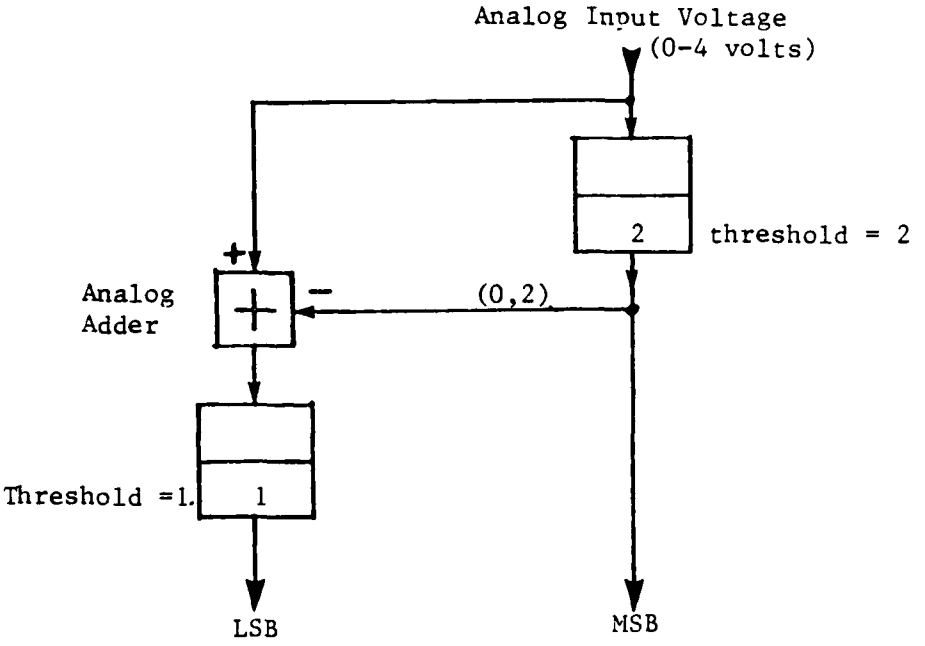


Figure 5. 2-Bit Threshold A/D

The converter functions by evaluating whether the input signal is larger or smaller than one MSB (Most Significant Bit) which is 2 volts in the figure. If one MSB fits into the input signal then the digital output comparator is high. In the next step a voltage equal to the MSB is subtracted

from the analog signal before it arrives at the next threshold logic element (comparator). This comparator determines if the remaining voltage is greater or less than 1 unit, which is the value of the next bit position in the digital output word. If the remaining voltage is greater than 1 then the output bit position is a high. Thus this A/D converter works by successive subtraction and it is evident in this 2-bit converter that the decision on the Least Significant Bit must wait until the Most Significant Bit is decided.

Figure 6 shows a 4-bit threshold A/D system, with the analog subtraction combined into the comparator. The analog comparators that perform addition and subtraction and have specified switching levels are called "threshold logic elements with weighted inputs." This converter also functions by first evaluating whether the input signal is larger or smaller than one MSB. If one MSB fits, then this voltage (8 in the figure) is subtracted and the next smaller bit is checked. This technique is known as successive approximation if only one comparator and one D/A, which does the subtraction, is used, and requires N steps for an N bit analog to digital conversion. The threshold A/D also works by successive approximation, however using a separate comparator for each bit decision and several D/A's to do the subtraction at the appropriate comparators. The system does not operate completely in parallel like a "flash" A/D but it is faster than a clocked successive approximation A/D. The clocked type is slower because extra space for timing errors must be made at each step of the N cycles. The threshold A/D is asynchronous and has the result "trickle down" with the digital output accepted after a suitable delay.

5.1.2.2 Accuracy Requirements

A. Noise and Spurious Signals

A contemporary analysis of A/D accuracy involves several kinds of tests which show the individual and collective effects of nonlinear and frequency dependent errors. For example, one easily performed test uses an analog sine wave as the input to the A/D, followed by a "perfect" D/A which then hopefully

reproduces the sine wave. Errors in the analog to digital conversion result in excess background noise, harmonics and other spurious signals which may be observed with a spectrum analyzer. Providing two sine waves at different frequencies simultaneously for the test signal will give intermodulation products and yield information about the A/D which is especially appropriate to applications having strong input signals.

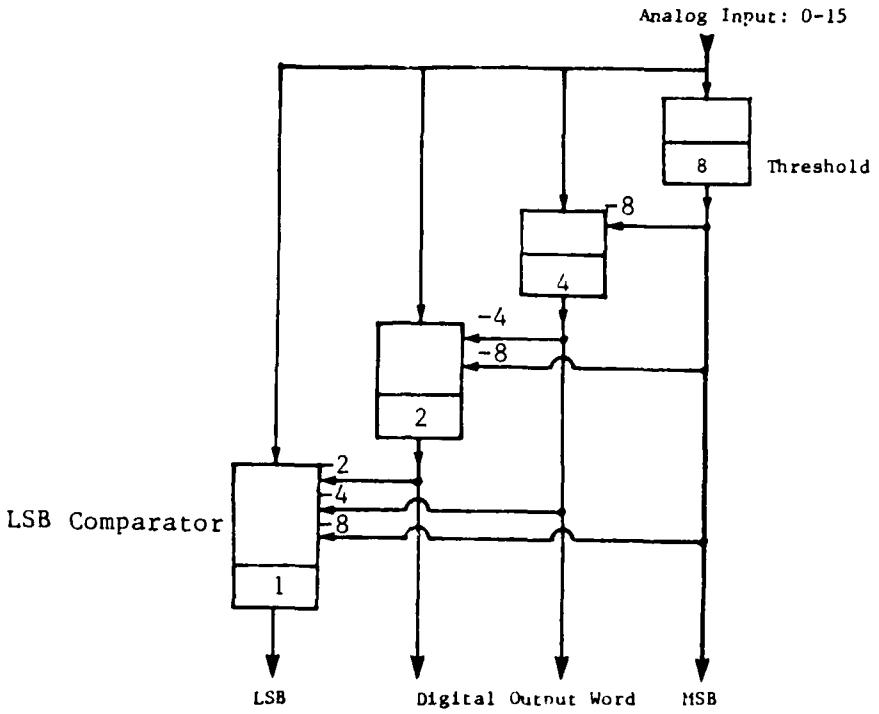


Figure 6. Threshold Logic A/D

A second test which is quite revealing gives the signal-to-noise or signal-to- distortion ratio for applications having wide bandwidth signals. In this case an analog input signal is also fed into the A/D and a "perfect" D/A reproduces the signal which is again observed with a spectrum analyzer. The input signal is not a sine wave however, but wideband noise (within aliasing requirements) with an empty notch. The spectrum analyzer reproduces this noise spectrum with the notch still evident, but unfortunately filled in somewhat. This revealing tests gives the "noise power ratio" (depth of the

notch in db) which is smaller than the signal-to-noise ratio obtained when the input signal was a simple sine wave.

B. Monotonicity

The basic accuracy requirement of the threshold A/D is explained below. This analysis gives a simple and useful accuracy result, but does not address the periodic nonlinearities which give rise to harmonics and intermodulation components.

There are two mechanisms which trigger the change in the digital output word as the input analog voltage changes. First, the LSB comparator can alone change state because the analog voltage crosses (in either direction) the threshold at this comparator. (Note that the actual threshold of the LSB comparator is modified by the state of the larger more significant bits.) The second case occurs when the changing analog input voltage crosses the threshold of any higher order comparator causing it to change state. Only one higher order comparator is triggered by the slightly changed input voltage, though that comparator then changes the thresholds on the lower comparators which causes further changes in the lower comparators. The point here is that only one comparator at a time is supposed to initially trigger the change if the input signal changes slowly, and that the LSB comparator alternates with the higher order comparators in triggering this change when the input signal is a slowly ramping voltage. As a result, the A/D error called missing codes can be excluded by requiring that the threshold of the LSB comparator and each of the other comparators never overlap with any combination of the D/A subtraction lines. This requirement is met if the offsets of each comparator in conjunction with the D/A subtract line feeding that comparator is less than 1/2 LSB.

The unwanted overlap of the thresholds of two higher order comparators would require a much larger error equal to 2 LSBs or 4 LSBs etc., which we will ignore here.

An N bit threshold A/D requires N-1 D/A converters each with ±1/4 LSB accuracy on the N bits. We assume ±1/4 LSB accuracy in the N comparators. This guarantees monotonic behavior with no missing codes.

5.1.2.3 Speed and Relative Complexity of the Threshold A/D Converter

The speed of the threshold A/D is the time required for N comparators to settle in tandem, plus the time for N-1 D/A converters to settle, also in tandem. However the N-1 D/A converter settling times are different. The first settling time involves a change of one MSB in the D/A. The second settling time involves only the next lower bit. Finally the last "D/A" settling time only involves a change of one LSB in the D/A. Thus the trickle down settling time required is much less than N-1 full D/A settling times. However, on this point one should note that common A/D's using the successive approximation technique also benefit from these decreasing D/A settling times as it steps through the conversion process.

This threshold A/D is faster than a successive approximation A/D, but is slower than a flash A/D. This threshold A/D has more parts than a recirculating successive approximation A/D, which needs one comparator, one D/A, and one digital counter. A flash A/D requires 2^N-1 comparators and a digital decoding circuit for an N-bit converter, but it generally does not require an analog sample-and-hold circuit at the input as is required by the successive approximation A/D and threshold A/D systems.

5.1.3 Sideways Summer

5.1.3.1 Introduction

In earlier work on digital optical computing, the process of converting the analog output from the Digital Multiply by Analog Convolution (DMAC) algorithm to a computer usable binary form has either been given little attention or simply ignored. In fact, this conversion process, typically performed by A/D conversion and then shifting and adding (S/A), can be the slowest part of an entire calculation. We show here a novel technique for

reducing the delay time from 2N-2 full adder delays to $\log_2$ (N-1) delays where N is the number of bits being multiplied.

5.1.3.2 DMAC Operation

Most digital optical computers base their computations on an algorithm that provides digital accuracy by using an analog operation. As an example, consider multiplying the two numbers 7 x 7 = 49 digitally:

```
Binary 7 =     111
             x 111
               111
              111
             111
            110001   binary
             12321   mixed binary
```

If we add up the partial products without the carry, the sum will have a number larger than what can be represented in binary. That is, the result will be larger than 1. This non-binary result is said to be in "mixed binary" form.

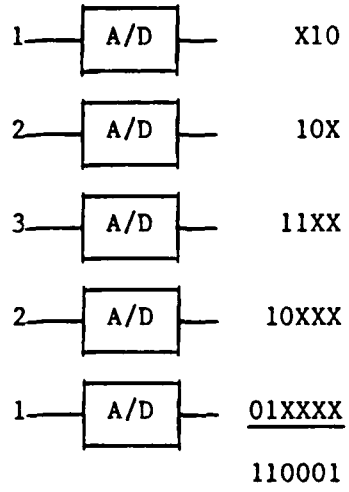
If we take the two sets of binary numbers and convolve them,

(111)*(111) = 12321

the output is exactly the mixed binary results of the previous example. To translate this result to the equivalent binary answer, each digit of the mixed binary analog number must undergo the following process:

1. Pass number through an A/D converter
2. Left-shifted one digit with respect to the previous partial product
3. Add the converted number to running sum.

This procedure is more clearly illustrated if the numbers of the example are represented in the following manner:



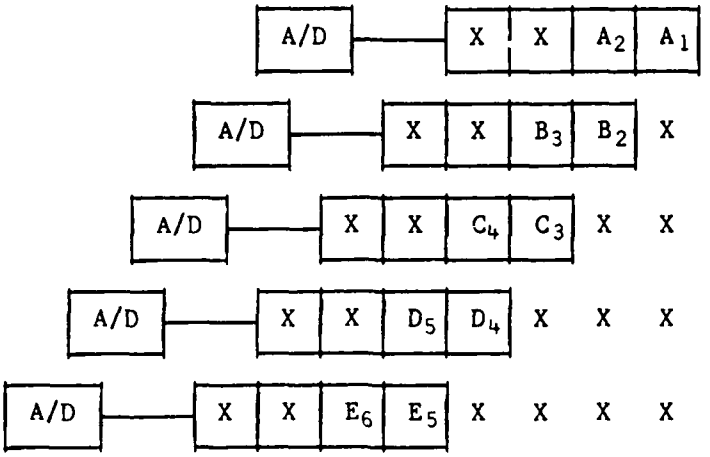
Note that the largest analog number is at most equal to the number of bits being multiplied (thus the use of 111 in the example). Also, the number of A/D converters needed to perform the mixed-binary to binary conversion is equal to 2N-1. Therefore, for an N-bit multiplier, one needs 1 $\log_2 N$ A/D', 2 log(N-1) A/D's, etc.

5.1.3.3 Implementation of the Adder

One of the simplest ways to perform the S/A is to use off-the-shelf (OTS) adders. Typically, OTS full adders are 4 bits in width, require about 10 ns to perform an add, 15 ns for the carry, and are commonly found with '83 suffixes (e.g., 7483, 74283, etc.). Note that there are also special purpose adders (accumulators) which incorporate shift registers (e.g., the SN74S281). However, these devices are slower to compute a sum than their conventional counterparts.

For ease of notation, represent each A/D required for the conversion from the set <A,B,C,D,E>, and each bit coming out of these A/D's by a number,

starting with '1', including the "don't care" place holding x's. For example, the bottom A/D will have bits E_7, E_6, ..., E_1. The following scheme to add the number may now be implemented:



This implementation requires a full adder for rows A and B, another for the partial product and row C, etc. In general, it requires 2N-2 delays to perform the conversion. If we implement a digital/optical system to multiply two 16-bit numbers, there would be 30 adder delays. Since typical delays are on the order of 7 nsec for, say, high speed CMOS, the resulting multiplier requires roughly 210 nsec to produce a binary answer. Given that an optical system can produce the mixed binary result in about 50-100 nsec, a potential bottleneck can occur as a result of the conversion.

5.1.3.4 Sideways Adder

If we take the least significant bit of each A/D, and form a word 2N-1 bits wide with the output from A/D converter A at the right and add this to the word formed by taking the next least significant bit of each A/D converter shifted left, etc. we see that we now have to add only $\log_2 N$ words that are 2N-1 bits wide.

X	E_5	D_4	C_3	B_2	A_1
E_6	D_5	C_4	B_3	A_2	X

The sums of each column are identical to those in the previous figure, but the propagation delay is substantially less. In fact, as the number of bits to be multiplied increases, the delay time to produce the binary output increases only logarithmically for the sideways summer as opposed to doubling for the conventional "shift and add" implementation.

A comparison of typical delays may be seen in the following table:

Summation Delay

Number Bits to be Multiplied	Conventional	Sideways
8	14	2
12	22	2.5
16	30	3
32	62	4

5.1.3.5 Implementation In Advanced Technologies

Full adders are extremely simple circuits. Typical 4-bit full adders use about 25 NOR-equivalent gates, most of which make up the "AND" and "XOR" functions. A 32-bit full adder, on a single chip and with fast carry, can be implemented with about 1000 devices. This circuit would require 32 input pins (plus 1-5 pins for power) and 32 output pins (plus 1-5 ground). Thus a minimum of 68 pins are required. A design incorporating 1000 gates is well within the grasp of present VLSI technology, and switching speeds of approximately 1 nsec/gate is possible. (There would be 3-5 gate delays between input and output.) Assuming 32 bit multiplication, with worst case 7 nsec delay per adder, the conversion time after the A/D would be 30 nsec.

5.1.3.6 Power Considerations

Since several different technologies are currently available, it is important to consider the tradeoffs between power and speed when implementing the Sideways Summer. For example, a 32-bit summer implemented in CMOS would typically require 50 mw of power and could be laid out in roughly 2000 μm^2. Note that typical delay times in current CMOS implementations are 1-3 nsec per gate.

Faster implementations may be realized with ECL or even GaAs implementations. Typical delay improvements of one order of magnitude can be expected. The tradeoffs include higher power (roughly three orders of magnitude for ECL) and larger area (again 1-2 orders of magnitude).

5.1.3.7 Conclusion

The potential bottleneck of converting "mixed binary" to binary can be reduced logarithmically. Furthermore, the building of a custom sideways summer in VLSI technology is well within current electronic technologies, amd it is reasonable to assume that since these circuits are common to all optical computers, that the Sideways Summer will be rapidly accepted.

5.2 ARI Multiply Accumulate Unit (MAC)

What has been presented in the past three sections is the workings of a replaceable electronic component; the MAC (see Figure 7). Because it accomplishes addition at a speed limited by the optical source, this device is potentially much quicker than its electronic counterpart. In fact, this advantage becomes dominant as the number of products to be added increases. Since it takes approximately 2 ns to create a mixed binary sum, the optimum number of products to be added is about equal to the (electronic) output conversion time divided by 2 ns. The best projected speed of the A/D is about

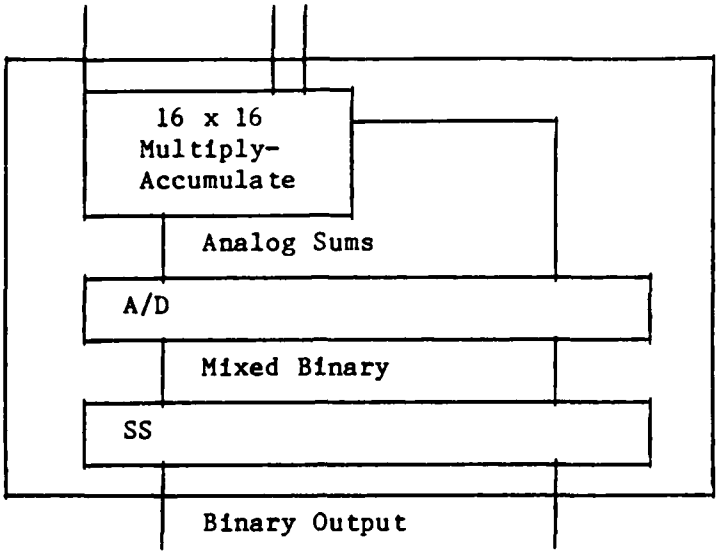


Figure 7. Diagram of Array Processor (MAC)

10 ns; the SS uses about 30 ns. This totals 40 ns. Therefore, 20 products should be accumulated. In contrast to this calculation to determine how many products should be added, a calculation is needed to determine the needed dynamic range of the system. The limiting factor for the dynamic range turns out to be the A/D converter, which is technology limited. The need for "flash" A/D converters was evidenced in the section on TLU A/Ds, and at present 6 bits running at 10 ns is the state of the art. This will undoubtedly improve in the future, for example to 8 bits, which produces 256 levels to decode. A sixteen bit convolving multiplier needs to sum 16 diagonals; therefore 16 products can be accumulated. This comes close to the suggested accumulation of 20 products.

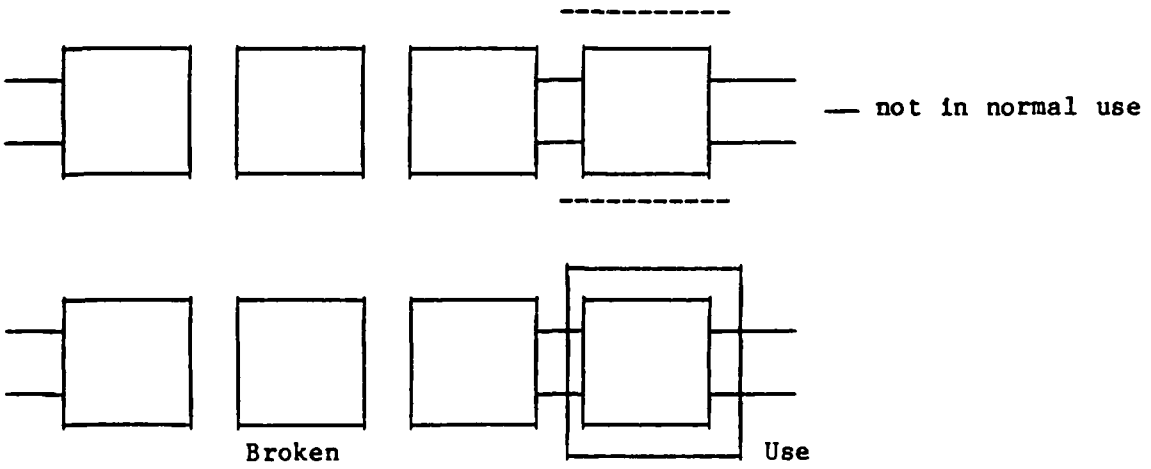
Before we explain how we could improve this system, a detailed specification would be helpful. We have created an asynchronous integrated optical multiply accumulate unit that performs 16 bit integer calculations. The device is built on an integrated optical substrate, probably $LiNbO_3$, possibly GaAs in the future. Included are 256 optical AND gates, 31 Flash A/D converters which follow a charge collecting circuit, with the 31 A/Ds leading

into a VLSI sideways summer that is made of high speed CMOS. Sixteen multiplications are created and summed every 80 ns with full individual 32 bit products. This implies a 200 MHz throughput rate, or to use a better performance indicator, 200 million multiply adds/sec (MAS). This chip by itself would be the heart of an array processor that performs 200 M MAS. No array processor runs that fast primarily because of slow addition, unless there is a high degree of parallelism (e.g. many MAC's). However, this one chip could conceivably multiply a 16 x 16 matrix by a 16 x 1 vector in 16•80 ns = 1.52 μs. If 16 of these MACs were run in parallel, the time required drops to 80 ns. Systolic architectures are discussed later.

5.3 Fault Tolerance

Because our device is a module, the problem of a malfunction can be solved by replacing the module. In contrast, consider the case of an N channel A.O. cell. Each channel performs a multiplication. N channels means N multiplications can be done simultaneously. If, however, one channel breaks, the entire N channel A.O. cell must be replaced. This amounts to replacing the processor, not a part of the processor. Consider the above simple example, but using the ARI module; performing the N multiplications simultaneously would now require N modules. If one module were to break, it could either be replaced, or another module could serve double duty. While the speed drops somewhat, the processor could continue to function.

If we consider the scenario where the mathematical problem to be solved involves a systolic architecture, the advantages of the modular approach become even more attractive. Using N+Y modules allows for Y units to fail. However, since the systolic architecture includes a pipeline, the number of modules in use will stay the same.



This cannot be as easily accomplished in the A.O. architecture, especially using the SAOBic architecture. The SAOBic architecture requires that each channel correspond to a vector. If one channel breaks, the whole vector cannot propagate through without errors. The only feasible solution for the bulk architecture to become fault tolerant is to 1) detect the bad channel, and 2) redirect the data from the bad channel to an extra "good" channel. This would preclude the use of the SAOBic architecture since each channel has a specific relation to its neighbor.

Using the ARI modular approach has introduced a technique to combat the fault problem which 1) is easier to implement than bulk optics, and 2) can take advantage of previous research on fault tolerance done for purely electronic processors.

See Appendix C for a quantitative analysis of fault tolerance in optical processors.

6. REFERENCE

1. P.S. Guilfoyle, Opt. Eng. 23, 20 (1984).

APPENDIX A

Once hardware has been developed, an algorithm that fits the specific architecture should be chosen. In the particular problem considered here, a set of simultaneous equations is to be solved with a specified accuracy and time requirement. Therefore, the algorithm must be optimized to meet these constraints.

Ideally, floating point computations on a direct matrix inversion algorithm (DMI) would provide a stable accurate result. Unfortunately, in this adaptive processing problem, the time constraint (solving the problem in 2-10 μs) precludes the use of floating point hardware. (This is not entirely true. The new Weitek and AMD floating point chip sets can run with 90 ns clocks, thereby allowing 10 multiplications per update time. If enough chips are used, the possibility of solving the problem in the allowed update time becomes closer to reality. Actually, results have shown that for this floating point hardware to be viable, 50 μs update times are the minimum that should be considered.) Nonetheless, the optical processor we have been describing uses only fixed point calculations.

Second, the MI is usually not the best choice for many considerations. (Various algorithms and their attributes will be described in Appendix B). Suffice it to say that the algorithm needs only to converge to a solution that is important to the problem. For example, if the jammer power is 40 dB greater than the signal, and the processor provides 41 dB of adaptation; it is not important that the real solution of 41.748 dB, which needs 2x more time, be obtained.

APPENDIX B
OPTICAL THRESHOLD LOGIC A/D CONVERTER

Conjugate Gradient Methods

Various forms of gradient methods have been investigated for use in solving systems of simultaneous linear algebraic equations

$$Ax \simeq b$$

The approaches minimize the square of the residual, $R = b - Ax$. The conjugate gradient method developed by Hestenes and Stiefel[1] offers the advantage of simplicity in application, an important consideration for optical implementation. In addition to computational simplicity the approach has three other major advantages

- The original matrix A is not modified during the computation. This insures that errors do not accumulate from step to step in the procedure. If A has many zeros, this feature allows the exploitation of the sparcity. The conjugate gradient approach is a method of choice for large sparse systems.
- At each step of the procedure the estimated solution vector improves and the square of the residual monotonically decreases. That is $||x-x_i||$ and $||R_i||$ monotonically decrease.
- The procedure can be restarted at any point.

These features have direct application in adaptive processing. If we have a solution x^k to $A^k x^k = b^k$ and A^{k+1} is formed from the addition and some rows and the deletion of others, the conjugate gradient approach can be started for the $A^{k+1}x^{k+1} = b^{k+1}$ with an estimated x which uses the knowledge of the previous solution. One can start the k+1 problem with the k^{th} solution. While the approach to adaptive processing is formally a batch procedure it is not necessary to ignore previous information when forming a

trial solution. A batch application of the conjugate gradient method has been successfully applied to the adaptive processing problem by Daniel.[2]

The basic algorithm for solving $Ax = b$ is as follows. Select a trial solution x_o then the initial residual R_o is found and the procedure starts.

$$R_o = b - Ax_o \qquad P_o = S_o = A^T R_o$$

$$q_i = Ap_i \qquad \alpha_i = ||S_i||^2 / ||q_i||^2$$

$$x_{i+1} = x_i + \alpha_i P_i \qquad R_{i+1} = R_i - \alpha_i q_i$$

$$S_{i+1} = A^T R_{i+1} \qquad \beta_i = ||S_{i+1}||^2 / ||S_i||^2$$

$$P_{i+1} = S_{i+1} + \beta_i P_i$$

At each step the square of the residual is smaller than the square of the residual of the previous step. $||R_{i+1}||^2 < ||R_i||^2$. The conjugate gradient technique formally converges in a finite number of steps and is in this sense a direct method. In the absence of roundoff, N iterations are required where N is the number of nonzero singular values of A. For ill-conditioned problems where roundoff can be expected, convergence is not reached in N steps. By treating the method as if it were an iterative one and continuing one convergence is typically obtained in less than 2N steps[3]. Unlike other iterative approaches, prior knowledge about the singular values of A is not required in order to establish convergence. Indeed the parameters α_i and β_i of the algorithm are often used as acceleration parameters in iterative methods.

The algorithm is equivalent to applying the conjugate gradient method directly to the normal equations, however as Hestenes and Stiefel[1] indicate, the algorithm is numerically superior to solution of the normal equations since A^TA is never explicitly formed. Elfving[4] has studied several

different formulations of the conjugate gradient method as applied to the ill-conditioned problem as has Reid.[5] The general consensus is that the algorithm gives accurate results.

The solution errors that are caused by solving the linear system of equations on a short word length computer (analog or optical) are magnified by the condition number of the matrix A. We have investigated the feasibility of using equilibration techniques to transform the problem to a system of less ill conditional equations.

$Ax = b$ can be transformed into $Hy = g$ by the identity

$$(TAS)S^{-1}X = Tb$$

where

$$H = TAS$$
$$y = S^{-1}X$$

and

$$g = Tb$$

T and S are nonsingular. For a square matrix the transformation TAS do not change the problem mathematically. However the condition number of H is not the same as that of A. The condition of H can be controlled by the selection of T and S. If we restrict T and S to be diagonal, then T scales the rows of A and S scales the columns of A.

The desired scaling is one which makes the rows and columns of H as linearly independent as possible. The optimal scaling will cause the condition number of H to be a minimum. This problem has been solved for certain matrix norms by Bauer.[6] However the solution depends on knowledge of A^{-1}. This is information that is not available a priori. A partial solution is given by Van Der Sluis.[7] He showed that for column scaling, T = 1, the

condition obtained from choosing S so that the columns are of unit length differs from the condition of the optimal S_{opt} by less than the square root of the dimension of A.

Major scaling will not only aid in operating convergence, but will reduce roundoff error as well. This is particularly important since the optical processor is not a floating point machine. If scaling is done by powers of the exponent, the mantissa of the numbers are not affected. Thus there should be no effect on roundoff errors due to diagonal scaling for floating point processors. This is not the case for fixed point processors where roundoff errors should be affected severely.

Scaling has the effect of decreasing the disparity in the sizes of the elements. A useful procedure is to select the scale so that the largest elements in each row and column are on the order of unity.

References

1. M.R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," J. Res. Nat. Bureau of Standards 49, 6, 1952.

2. S.M. Daniel, "Batch Covariance Relaxation (BCR) Adaptive Processing," SPIE 298, 165 (1981).

3. A. Bjorck, "Methods for Sparse Linear Least Squares Problems," in Sparse Matrix Computations, Ed. J.R. Bunch and D.J. Rose. Academic Press, New York (1976).

4. T. Elfving, "On the Conjugate Gradient Method for Solving Linear Least Squares Problems," LITH-MAT-R-1978-3, Linkaping (1978).

5. J.K. Reid, "On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Linear Equations," 231-254, in Large Sparse Sets of Linear Equations, J.K. Reid, Ed., Academic Press (1971).

6. F.L. Bauer, "Optimally Scaled Matrices," Numerische Mathematik 5, 73 (1963).

7. A. Van Der Sluis, Nemer. Math. 14, 246 (1970).

APPENDIX C
FAULT TOLERANCE AND SELF HEALING IN OPTICAL SYSTOLIC ARRAY PROCESSORS

C.1 Background

To our knowledge all optical systolic array processors so-far described work only if every component part works perfectly.[1-8] Yet these computers involve many hundreds of sources, modulators, detectors, and convertors, D/A convertors, etc. Many of these must operate at both very high speed and very high accuracy. Drifts in speed or accuracy which might be tolerable in other applications may have disastrous effects in such a computer. Rather than rely on "perfection," we believe it wiser to allow for failures. The consequences would be lower cost systems (higher yield) and more reliable operation.

In most other fields, as well, fault tolerance has proved necessary and desirable, so a large body of knowledge concerning general approaches has been developed. Here we attempt to apply some of those approaches to optical systolic array processors.

C.2 General Approaches

We feel compelled to remark that good design, good component choice, etc. can reduce the failure rate. It simply can not be asked to achieve defect-free construction and operation at all times. All other remarks must be tempered by this one. We discuss two general approaches to error control below.

The first widely-used method is damage containment. In optical systolic array processors, this principle favors smaller modules, so construction yields can be high and replacement costs can be low. On the other hand, construction from discrete components is obviously impractical and uneconomical. Arrays of sources, modulators, detectors, etc. seem desirable. Thus each case involves a tradeoff and general conclusions are not possible.

The second general approach is to "heal" defects when they are detected. This is not always possible, so we must divide defects into two classes:

healable and nonhealable or, as we prefer, minor and serious. Note that healing may not be an infinitely-repeatable process, so a defect which was minor in its first n occurrences might be major in the (N + 1) st occurrence.

The third general approach is to "bypass" serious defects when possible. When a serious defect can not be bypassed, the system or module having that defect must be replaced. This brings us back to the containment principle.

We will concentrate on the healing and bypassing strategies for optical systolic array processing, but we must precede those discussions with remarks on defining and detecting defects.

C.3 Defining and Detecting Defects

For our purposes, defect definition is easy. A defect in any component is the operation of that component in a manner that causes an erroneous result to occur in the calculated results.

In detecting such a defect we are driven (by the defect definition) to make diagnostic calculations and compare the results with the known correct results. If the diagnostic calculations are chosen well, we can at-least-partially isolate the defect.

For the sake of simplicity we will confine our search call "individual multiply-accumulate channels." Furthermore, again for simplicity, most of our remarks will be directed toward Matrix-Vector multipliers. Figure 1 shows, at least symbolically, how a group of N channels can multiply an NxN matrix by an N component Vector. In practice these channels can be constructed in many ways: modulated source arrays, acousto-optic devices, surface acoustic wave devices, discrete channel integrated optics, one source with discrete modulators, etc. These details are important to the system design but not to our discussion on diagnostics.

Diagnostics must begin with a model of the channel. We present and use a simple model here. The designer must either make his components conform

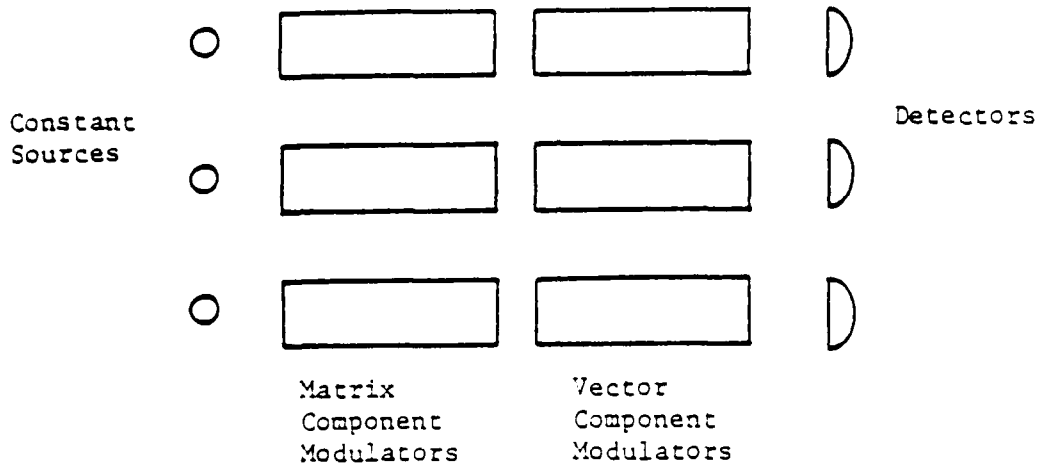


Figure 1. A Symbolic Representation of An Optical Systolic Array Processor. In practice the sources may be modulatable, the vector modulation may be via an acousto optic cell, etc. Nevertheless, distinct channels can always be defined.

(sufficiently-well so as not to change calculated results) with this model or use a more appropriate model. We suspect the former approach will be easier.

Our channel model will assume

- o A controllable-intensity light source set slightly below its maximum
- o A matrix component modulator, with a transmission T, which depends on its control voltage V, according to

$$T_1 = T_0^{(1)} + \left[T_m^{(1)} - T_o^{(1)}\right] (V_1/U_1) \quad , \qquad (1)$$

where

T_o = minimum transmission ,

T_m = maximum transmission ,

and

U_1 = reference voltage ,

- A Vector component modulator with transmission dependent on its control voltage V_2 according to

$$T_2 = T_0^{(2)} + [T_m^{(2)} - T_0^{(2)}] (V_2/U_2) \quad , \tag{2}$$

where the quantities $T_0(2)$, $T_m(2)$, and U_2 are defined as in Eq. (1).

- A detector whose ideal output will be defined as 1 when $V_1 = U_1$ and $V_2 = U_2$.

Let us write the total output in terms of

$$R_1 = V_1/U_1 \tag{3}$$

and

$$R_2 = V_2/U_2 \quad . \tag{4}$$

We have, omitting some uninteresting proportionalities constants and assuming a properly-adjusted source brightness, a signal

$$S\ (R_1,R_2) = T_1(R_1)\ T_2(R_2) = \{T_0^{(1)} + [T_m^{(1)} - T_0^{(1)}]\ R_1\}$$

$$\times\ \{T_0^{(2)} + [T_m^{(2)}\ T_0^{(2)}]\ R_2\} = T_0^{(1)}\ T_0^{(2)}$$

$$+ T_0^{(1)}\ [T_m^{(2)} - T_0^{(2)}]\ R_2 + T_0^{(2)}\ [T_m^{(1)} - T_0^{(1)}]\ R_1$$

$$+ [T_m^{(1)} - T_0^{(1)}]\ [T_m^{(2)} - T_0^{(2)}]\ R_1\ R_2 \quad . \qquad (5)$$

With only a little more uninteresting algebra, we find

$$S\ (R_1,R_2) = S\ (0,0) + [2S\ (1,1/2) - S\ (1,1)]\ R_1$$

$$+ [2S\ (1/2,1) - S\ (1,1)]\ R_2$$

$$+ [3S\ (1,1) - 2S\ (1,1/2) - 2(1/2,1) + S(0,0]\ R_1R_2 \ . \qquad (6)$$

That is, by measuring four outputs [S (0,0), S (1/2,1), S (1,1/2), and S (1,1)] we can determine the current operating curve S (R_1,R_2).

Now we must address the question of whether the S (R_1,R_2) function just measured is acceptable. We will use as criteria

$$(1 - \varepsilon) < |S\ (1,1)\ | < (1 + \varepsilon) \qquad (7)$$

and

$$S(0,0) < \varepsilon \quad . \tag{8}$$

The value of ε depends on the needed channel dynamic range. If the dynamic range D is needed, a reasonable choice would be

$$\varepsilon = 1/(2D) \quad . \tag{9}$$

Other criteria might be used, but this one is simple,probably adequate, and easy to test. A channel failing either test is declared to contain a defect or that the channel has failed.

As an aside, note that the more our system design increases D, the more frequently a defect will occur given the same components.

C.4 Healing A Failed Channel

Because channels can fail in two ways (Expressions 7 and 8), two methods of healing must be available. A channel which fails the criterion of Expression 7 will be called "miscalibrated." A channel which fails the criterion of Expression 8 will be called "noisy."

A miscalibration can be corrected by adjusting some combination of source brightness and detector gain. Note that recalibration of a channel in which $S(R_1,R_2)$ was less than unity could cause the channel to become noisy. Thus Expression 8 must be retested after an upward recalibration. If Expression 8 is not satisfied, the channel has a serious failure.

A minor noise problem is one which can be removed by adjusting downward some combination of source brightness and detector gain while still satisfying Expression 7. If such an adjustment is impossible, the channel has a serious failure.

C.5 Bypassing Channels With Serious Failures

We assume a module needing N operating channels to perform. To allow for F channel failures per module we will need N + F channels per module. Of course

$$F \ll N \quad . \tag{10}$$

We will assume that the N Matrix-information lines can be reconfigured slightly and that up to F clock time delays can be inserted anywhere before, within, or after the string of N vector components initiating the calculation. The rest is easy and will be illustrated with the special case of N = 3 and F = 1, The Processor has N + F = 4 channels. The problem we want to solve is the evaluation of

$$\vec{y} = \vec{A}\vec{x} \quad = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{11}$$

If the first three channels are good, we solve the augmented problem.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 0 \end{bmatrix} \tag{12}$$

If, for example, the second channel is seriously failed, we solve the augmented problem.

$$\begin{bmatrix} y_1 \\ 0 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & a_{12} & a_{13} \\ 0 & 0 & 0 & 0 \\ a_{21} & 0 & a_{22} & a_{23} \\ a_{31} & 0 & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ 0 \\ X_2 \\ X_3 \end{bmatrix} \tag{13}$$

The generalizations to matrix-matrix processors and to F > 1 are obvious. The final results may have to be buffered for up to F clock periods if we want to guarantee the output temporal format.

C.6 Extension To Digital Calculations

Those familiar with digital optical systolic array processors will note that the situation becomes more complicated when the numbers are represented with a string (spatial or temporal) of signals. We may be using a twos complement representation of 16 bit real numbers. This requires 17 channels to replace each of the individual modulators in the systems described earlier. To bypass F channels per multiplier unit, we need to allow 17 + F channels per multiplier unit. The complication arising from this approach is now easy to explain. The matrix coefficient channels run independently, but the vector coefficient channels intercommunicate. That is, there is no built-in bypassing mechanism in any module. If a serious failure occurs in the third channel of any multiplier, then there must be a zero as the third component. If we group M multiplier units into a module, we need MF spare channels to heal F serious failures. As the last step of any module we can make a smart buffer system which can reorder the vector component data in the form needed for the next module. In a typical acousto-optic cell allows about

M = 60 17 channel multiplication units with 17 channels (60 x 17 = 1020). Allowing a 1% serious failure rate, we expect about 10 serious failures among the 1000 channels. If we make the very optimistic assumption that we will have at most one serious failure per multiplier unit, we must allow 27 (= 17 + 10) channels per multiply. This reduces M to about 39 (39 x 27 = 1026). More importantly, the time required for calculation increases by a factor of 27/17 (≈160%). If we go to an M = 1 module system, we need 18 channels per multiplier. The time per calculation increases by a factor of only 18/17 (≈ 6%). Of course this does not take into account the time lost in the smart buffer. Nevertheless this analysis suggests that high M modules such as those using 1000 pixel acousto-optic cells can be made fail safe against serious failures only at some considerable loss of speed and capacity.

C.7 Software Error Checks

The "checksum" method of Huang and Abraham[9] adds a bottom row to the A matrix and a rightmost column to the B matrix in performing

$$A \times B = C \tag{14}$$

The resulting C matrix has both the extra bottom row and the extra right column. The checksum rows and columns are simply projections along the orthogonal direction. The product matrix C has both row and column checksums which should also be correct. Thus error detection is possible. When there is a single error it can even be located and corrected.

Extending all of these properties to matrix-vector multiplication is not trivial. One way to do at least the error detection is to add both a bottom checksum row and a right column (all zeros) to the starting matrix A to form an augmented matrix A'. We then add a zero as the last component of $\vec{x}$ to form an augmented vector $\vec{x}$. This is best illustrated with the simple case

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{15}$$

$$\vec{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{16}$$

$$A' = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ S_1 & S_2 & 0 \end{bmatrix} \tag{17}$$

and

$$\vec{X}' = \begin{bmatrix} X_1 \\ X_2 \\ 0 \end{bmatrix} , \tag{18}$$

where

$$S_1 = a_{11} + a_{21} \tag{19}$$

and

$$S_2 = a_{21} + a_{22} \tag{20}$$

$$A\vec{x}' = \begin{bmatrix} y_1 \\ y_2 \\ y_1 + y_2 \end{bmatrix} \tag{21}$$

In this way we can detect but not correct errors. The trouble is that for large matrices the checksums may get too large for a fixed point computer to handle.

To counteract this we replace S_i with αS_i, where α is chosen to keep all of the $|S_i|$'s in range. We may choose α as a prescribed fraction of the maximum $|S_i|$ for each matrix, for instance.

C.8 Overall Approach

We can summarize our approach as follows:

1. Try to make highly reliable, predictable channels,
2. Automatically test each channel on a regular basis,
    - o When possible, heal any failures,
    - o Bypass up to F failures using the methods described, and
    - o Replace or bypass modules in which the number of serious channel failures exceeds F, and
    - o Use weighted checksums for real-time error detection.

The concept of bypassing modules is a simple extension of the concept of bypassing channels and is another possible benefit of using a modular approach to the construction of optical systolic array processors. Small modules have distinct when we are dealing with digital accuracy computations.

References

1. H.J. Caulfield, W.T. Rhodes, M.J. Foster, and S. Horvitz, Opt. Comm. 40, 86 (1981).

2. R.A. Athale and W.C. Collins, Appl. Opt. 21, 2089 (1982).

3. D. Casasent, J. Jackson, and C. Neuman, Appl. Opt. 22, 115 (1983).

4. R.P. Bocker, H.J. Caulfield, and K. Bromley, Appl. Opt. 22, 804 (1983).

5. R.A. Athale, Proc. 10th Int. Opt. Comput. Conference (1983).

6. P.S. Guilfoyle, Opt. Eng. 23, 20 (1984).

7. D. Casasent, Proc. IEEE, (July 1984).

8. W.T. Rhodes and P.S. Guilfoyle, Proc. IEEE, (July 1984).

9. K.H. Huang and J.A. Abraham, IEEE Trans. Comp. C-33, 518 (1984).

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.

END

FILMED

1-86

DTIC